VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2016 Tran Hoang An

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Lokalizační subsystém pro nemocniční mobilní robot

Localization subsystem for hospital mobile robot

2016

Tran Hoang An

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **An Tran Hoang**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Lokalizační subsystém pro nemocniční mobilní robot
Localization Subsystem for Hospital Mobile Robot

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Metody pro lokalizaci metodou simultánní lokalizace a mapování (SLAM).
2. Metoda založená na korelacích.
3. Implementace metody založené na korelacích v jazyce C.
4. Otestování a verifikace.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] KONEČNÝ, Jaromír. *Principy řízení mobilních servisních robotů*. Ostrava, 2014. Disertační práce. VŠB – Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra kybernetiky a biomedicínského inženýrství.
[2] VAN SICKLE, Ted. *Programming microcontrollers in C*. 2nd ed. Eagle Rock, Calif.: LLH Technology Pub., c2001, 454 s. ISBN 18-787-0757-4.
[3] GE, S a Frank L LEWIS. *Autonomous mobile robots: sensing, control, decision-making, and applications*. Boca Raton, FL: CRC/Taylor, 2006, xxi, 709 p. 22. ISBN 08-493-3748-8.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jaromír Konečný, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 29. dubna 2016 …….

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 …….

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu bakalářské práce panu Ing. Jaromírovi Konečnému, Ph.D. za jeho vedení, rady a v neposlední řadě podporu při této práci. Především děkuji za trpělivost a rady při implementaci algoritmu do jazyka *C*.

Rád bych poděkoval také své přítelkyni a rodině za podporu při tvorbě této bakalářské práce a to především vytvořením pohodlné atmosféry při práci.

Abstrakt

Tato bakalářská práce se zabývá praktickou implementací algoritmu založeném na vzájemné korelaci, který byl navržen pro lokalizaci automatizovaného robota v reálném prostoru. Tato práce nejprve popisuje známé algoritmy, které jsou v této době používány pro lokaci robota. Zabývá se nadále výhodami a nevýhodami existujících algoritmů. Rozboru je také podroben referenční algoritmus této práce. Poté se práce věnuje praktické části a to implementací algoritmu do jazyka *C*. Poslední část práce se zabývá experimentálním měřením hotového algoritmu. Následuje zpracování dat a závěr, kde jsou konfrontovány výsledky.

Klíčová slova: Zarovnání laserového snímku, SICK LMS 100, algoritmus lokace, korelace, laserový skener, C

Abstract

This bachelor thesis is mainly about practical implementation of algorithm based on cross correlation which was deployed for location of automatic robot in real environment. This thesis firstly describes known algorithms, which are being used for location of robot. It describes the pros and cons of each algorithm as well. The reference algorithm is a subject of analysis too. The work is then describing practical part, which is implementation to programming language. The last part of work handles the experimental measures of implemented algorithm. Follows by data processing and the conclusion where the results are confronted.

Keywords: Aligning laser lidar, localization, algorithm, ICP, laser scanner, SICK LMS 100, Matlab, C

Obsah

Seznam použitých zkratek a symbolů

$\mathbf{A}$	–	obecná matice bodů
APR	–	Kotvící body {Anchor Point Relationship}
ASCII	–	Americký Standardní Kód pro Výměnu Informací {American Standard Code for Information Interchange}
CPU	–	Centrální procesorová jednotka {Central Processing Unit}
ETX	–	Netisknutelný znak pro konec {End of text}
GPS	–	Globální družicový polohový systém {Global Positioning System}
$\mathbf{H}(n)$	–	hessián účelové funkce
ICP	–	Iterativní nejbližší body {Iterative Closest Point}
IP	–	Internetový protokol {Internet Protocol}
$\mathbf{M}$	–	matice množiny bodů
$\mathbf{M}_{t+1}$	–	matice nové množiny bodů
MCL	–	Monte Carlo Lokalizace {Monte Carlo Localization}
NDT	–	Normal Distributions Transform
STX	–	netisknutelný znak pro start {Start of text}
SLAM	–	Self Localization And Mapping
f	–	Cílová (účelová) funkce
$\mathbf{g}(\mathbf{x}_n)$	–	gradient účelové funkce
$\mathbf{i}_x$	–	souřadnice bodu na ose x
$\mathbf{i}_y$	–	souřadnice bodu na ose y
$\mathbf{j}_x$	–	souřadnice bodu na ose x
$\mathbf{j}_y$	–	souřadnice bodu na ose y
$\mathbf{m}$	–	afinní transformace
$\mathbf{m}_i$	–	vektor bodu množiny $\mathbf{M}$
$\mathbf{m}_x$	–	translace v ose x
$\mathbf{m}_y$	–	translace v ose y
r	–	laserový snímek
$textbfr$	–	vzdálenost [mm]
s	–	laserový snímek
$\mathbf{t}$	–	vektor rotace
t_x	–	parametr transformace v ose x

t_{x0}	–	počáteční podmínky translace v ose x
t_y	–	parametr transformace v ose y
t_{y0}	–	počáteční podmínky translace v ose y
x	–	souřadnice v ose x
x_i	–	souřadnice bodu v ose x
$\mathbf{x}_i$	–	vektor stavových proměnných
$\mathbf{x}_j$	–	vektor stavových proměnných
y	–	souřadnice v ose y
y_i	–	souřadnice bodu v ose y
α	–	délka kroku
ϕ	–	úhel natočení (rotace) [rad]
ϕ_0	–	počáteční úhel natočení (rotace) [rad]
τ	–	pomocná substituční proměnná pro korelaci
ω	–	úhel natočení snímku
$\star$	–	křížová korelace
0	–	index počáteční fáze
i	–	index i-té fáze
n	–	index n-té fáze
x	–	index pro osu x
y	–	index pro osu y
ϕ	–	index úhlu ϕ

1 Úvod

Uplatnění automatizovaných robotů a robotických systémů v medicíně je v dnešní době velice diskutovaným tématem. Motivací v zavádění těchto systémů je několik, mezi ně patří minimalizace lidských chyb, telemedicína a v neposlední řadě zefektivnění úkonů ve zdravotnictví. Tato bakalářská práce se bude zabývat algoritmem lokačního systému, který může mít řadu využití v oblasti medicíny a zdravotnictví. Jedním z mnoha příkladů je automatizované podávání léků, přípravků či dokonce přinesení novin pacientům. Při těchto úkonech je důležité, aby se robotický systém dokázal lokalizovat v mnohdy spletitých chodbách nemocnice a správně se navigovat do pacientova pokoje. Tímto by se značně ušetřil čas a práce personálu nemocnice a jak již bylo zmíněno dříve, eliminovala by se lidská chyba při podávání léků, která má mnohdy fatální následky.

S pokračujícím výzkumem by postupně úkony robotů mohly víc a víc nabývat komplexnosti, například výměna léků podávané infúzí. V budoucnu, pokud výzkum a inovace senzoru systému markantně pokročí, bylo by možné na základě podobného algoritmu sestavit systém snímání povrchů v lidském těle. Tento lokační systém by využívali například nanoboti, kteří jsou také velice diskutovaným tématem. Nanobot by snímal například miniruptury cév či vykreslování povrchu podezřelých objektů v tělě, například nádory. Tyto údaje dodané přímo z místa by obsahovaly cenné informace a data pro další postup v léčbě a rozhodování.

Celá tato práce se dělí na 4 kapitoly. První kapitolou je úvod. V Kapitole 2 se práce věnuje cílům, které byly stanoveny. Kapitola 3 je rozbor metod zpracování laserových snímků. Tato kapitola se věnuje existujícím metodám pro laserové zarovnání snímků, které se využívají v lokačních algoritmech. Poté se práce věnuje realizaci a implementaci vybraného algoritmu v praxi, což je obsahem kapitoly číslo 5. Předposlední kapitolou je experimentální část práce. Tato pasáž hodnotí a popisuje průběh měření, zpracování dat a následný výsledek, který z tohoto zpracování plyne. Výsledky implementovaného algoritmu jsou konfrontovány s výsledky referenčního algoritmu. Poslední kapitolou této práce je závěr, kde jsou zhodnoceny cíle této práce.

2 Cíle

Pro dosažení cílů této bakalářské práce je nutné pochopení již existujících metod zarovnávání laserových lidarů, které byly navrhnuty a realizovány. Tato práce se bude detailně věnovat a analyzovat navrženou metodu. Motivací je implementace navržené metody a následně její implementace na mikrokontrolery. Účelem této práce je implementace algoritmu do jiného programovacího jazyka pro snížení časové náročnosti výpočtu korelace. Dalším cílem je také připravit algoritmus pro metodu, která dokáže vytvářet mapu prostředí (SLAM), ve kterém se systém nachází a prozkoumává.

3 Rozbor metod pro zpracování laserových snímků

Lokační systém a orientace v prostředí byl a bude důležitou součástí každého automatizovaného robotického systému. S rostoucí lidskou infrastrukturou a potažmo rostoucí rozlohou měst, je obzvláště nyní potřeba přesně a rychle určit pozici v uzavřených prostorách místnosti.

V dnešní době je tato oblast velice diskutovanou a řešenou problematikou. Lokalizace a metoda zjišťování pozice je jednou z primárních dovedností robotického systému. Lokaci lze provádět pomocí různých senzorů, které již existují na trhu. Jeden z nejznámějších způsobů, kterým se budu ve své práci zabývat, je způsob pomocí laserových paprsků. Mezi další způsoby se řadí například metoda pomocí ultrazvukových dálkoměrů, odometrie[1], wifi lokace a v neposlední řadě nejrůznější GPS moduly.

Laserový snímač (dálkoměr) měří vzdálenost mezi daným snímačem (rozumějme robotem) a daným objektem v okolí, který měří okolí s úhlovým rozsahem od 180 ° do 360°. Krok snímání jednotlivých bodů je 0,25 ° nebo 0,5 °, kdy vrací zpět údaje o jednotlivých bodech objektů v okolí. V této kapitole se budeme zabývat jednotlivými existujícími typy algoritmu lokačního systému, konkrétně mapováním. Následující kapitola se zabývá metodami ICP, Monte Carlo, kotvícími body a metodou pomocí histogramů.

3.1 Metoda ICP

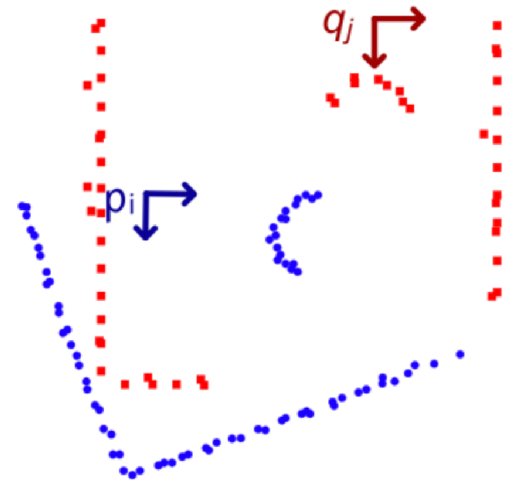
Metoda ICP (Iterative[2] Closest Point) je algoritmus, který funguje na základě hledání nejbližších dvojic bodů. Navržen byl matematiky McKayem a Beslem a myšlenka tkví v minimalizaci rozdílů součtů pravidelného čtyřúhelníku mezi určenými soubory bodů. Způsob minimalizace staví na předchozím vytvoření dvou bodů s nejmenší vzdáleností mezi sebou, z referenčního a aktuálního skenu. Tento algoritmus je nesčetně využívaným způsobem pro zjištění rotace a translace autonomních robotických systémů. Naměřené hodnoty mohou být totiž použity k vytvoření reálného obrazu zobrazujícího objekty v blízkosti robotického systému, které jsou tvořeny na obrazu jako sady bodů vykreslující reálné překážky a objekty.[2] Pro algoritmus jsou zapotřebí dva snímky, které byly zmíněny výše. Výsledkem porovnání snímků je rotace a translace aktuálního skenu vůči referenčnímu skenu. Nevýhodou ICP algoritmu je časově náročná výpočetní doba. Metoda ICP má tyto kroky: [3]

1. Předzpracování (preprocessing)

[1] Metoda, kdy přístroj snímá otáčky kola robotu

[2] Průběh děje minimalizace intervalu mezi dvojicí bodů se opakuje, dokud se nesplní některé podmínky pro ukončení algoritmu.

2. Srovnávání (matching)
3. Odmítnutí (rejecting)
4. Určení hodnoty cílové funkce
5. Minimalizace (minimazation)



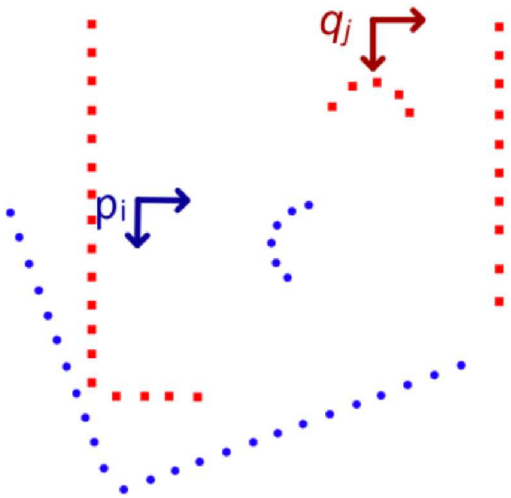
Obr. 1 Nasnímaný sken [22]

3.1.1 Předzpracování

Na obrázku 1 vidíme původní snímek. Počátečním krokem algoritmu je krok předzpracování. Tento krok eliminuje (filtruje) body, a to z důvodu redukce velikosti dat souboru. Takové body jsou například vzdálené body, které nejsou součástí algoritmu při určení lokace robotu v prostředí. Dále to jsou body, které se nacházejí vedle sebe ve velmi malé vzdálenosti. Pro názornost je přiložen obrázek 2.

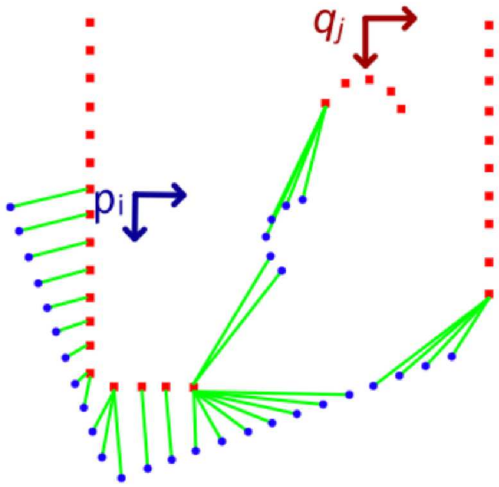
3.1.2 Srovnávání

Dalším krokem je výše zmíněné srovnávání (matching). Tato část algoritmu je však časově nejnáročnější. V tomto kroku jsou vyhledávány korespondující body mezi aktuálním a referenčním snímkem. Nejjednodušší způsob, jakým jsou vyhledávány je například hledání hrubou silou, (brute force) [1]. K výpočtu vzdálenosti se vybere bod, který je z aktuálního



Obr. 2 Předzpracování [22]

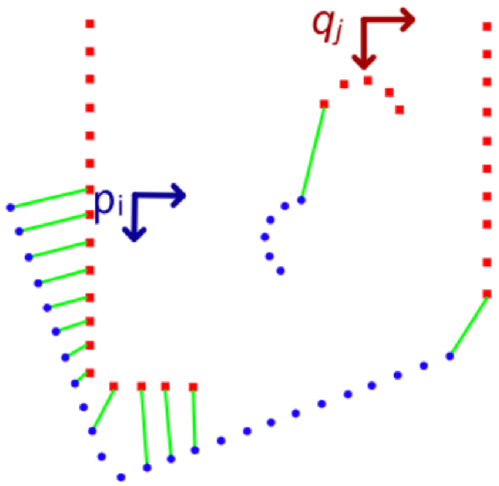
snímku nejblíž. Velkou nevýhodou tohoto způsobu zpracování ICP algoritmu je složitost a neefektivnost výpočtu. Díky těmto nevýhodám se metoda hrubou silou v praxi moc nevyužívá.



Obr. 3 Srovnávání [22]

3.1.3 Odmítnutí

Ve třetím kroku se provádí čištění párů bodů referenčního a aktuálního skenu. Tento krok je prováděn na základě výpočtu vzdálenosti mezi těmito dvěma páry. Do další části vstupují tedy jen páry bodů s nejmenší vzdáleností.



Obr. 4 Odmítnutí [22]

3.1.4 Určení hodnoty cílové funkce

V tomto kroku se vypočítá hodnota účelové funkce mezi páry bodů aktuálního a referenčního snímku. Hodnota funkce je zmenšována s každým opakováním algoritmu. S výslednou hodnotou se pak následně pracuje u minimalizace.

3.1.5 Minimalizace

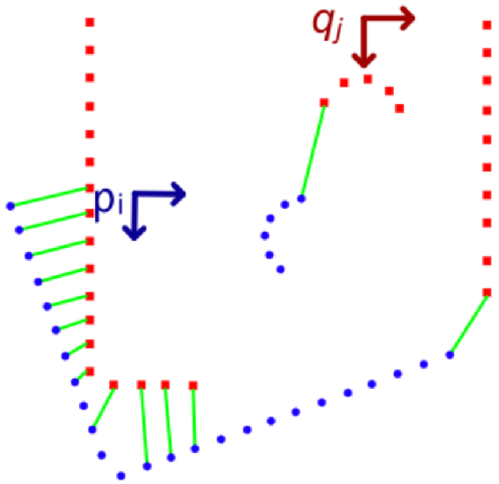
V tomto posledním kroku iteračně (opakovaně) realizujeme minimalizaci výsledné funkce. Pro minimalizaci součtů rozdílů čtverců mezi aktuálním a referenčním skenem potřebujeme vypočítat $f\,(t_x, t_y, \phi)$, což jsou parametry transormační funkce. Tyto parametry můžeme určit například pomocí Newtonovy metody [5].

$$p_{n+1} = p_n - \alpha.[\mathrm{H}_n(p_n)]^{-1}.g(p_n), n > 0 \tag{1}$$

kde α je délka kroku (u standartní metody je α rovna 1)

$g(p_n)$ je gradient v n–tém opakování

H_n je vypočítaný Hessián v n–tém opakování



Obr. 5 Minimalizace

3.2 Monte Carlo lokace

Kapitola o Monte Carlo lokalizaci bude v této bakalářské práci popsána velice stručně. O metodě Monte Carlo lokace (MCL) je již napsáno mnoho publikací [6, 7, 8]. Je hojně využívána roboty pro lokalizaci použitím částicového filtru.[9] Základní myšlenkou tohoto typu algoritmu je prezentace pravděpodobnosti stavu okolí výslednou množinou vážených bodů. Šance výskytu robotu v určitém místě je tedy určena váhami bodů a jejich zastoupením na daném místě. Se zvyšující se četností v okolí robotického systému se přesnost zvyšuje. Primární úlohou MCL je modifikace četnosti těchto bodů. Množina těchto bodů pak určuje pozici robotu. Monte Carlo lokace má čtyři kroky: [10]

1. *Predikční krok* - Přemísťováním robotu dostáváme novou množinu bodů M_{t+1} tak, že ke každému bodu m_{jt} z množiny M_t přidáme novou hodnotu vektoru proměnných $x_{i(t+1)}$. Po predikčním kroku se však zvyšuje nejistota pozice robotu.

2. *Translační krok* - V tomto kroku pracujeme se změnou lokace robotu. U translačního kroku ovšem pohyb nahradíme vstupem dat ze senzoru.

3. *Korekční krok* - Následně je v tomto kroku zpracováván údaj, který jsme získali ze senzoru. V této části je upravována četnost jednotlivých bodů. Nepracujeme zde přímo s vektorem x_j.

4. *Převzorkování* - V posledním kroku algoritmu získáváme ze stávající množiny bodů novou množinu bodů a to vyvozením. Tato nově získaná množina zahrnuje body se stejnou četností jako množina předešlá.

3.3 Plošné zarovnávaní snímků pomocí kotvících bodů

Další metoda, která umožňuje lokalizaci robotu se jmenuje metoda kotvících bodů (*APR - anchor point relationship*). Princip APR algoritmu funguje na základě komparativní metody mezi aktuálním skenem a větším množstvím referenčních snímků. Pro srovnávání jsou použity tzv. kotevní body. Pokud se v okolí nevyskytuje dostatečné množství těchto bodů, nelze zarovnání určit. [11, 5]

Kotvící body jsou podle [11]:

1. Virtuální rohy
2. Skokové rohy
3. Úhlové rohy

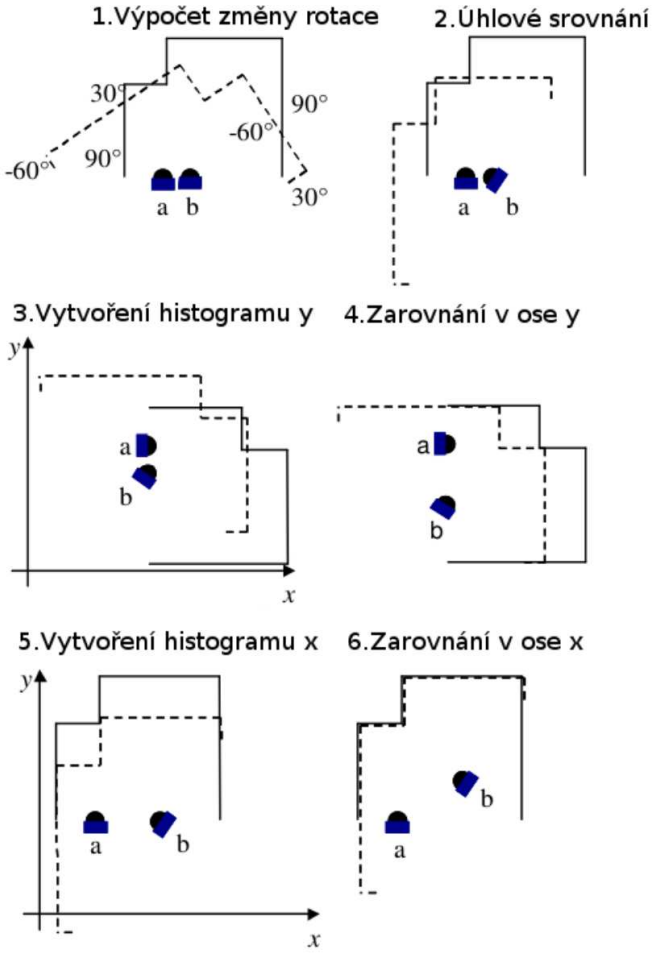
Virtuální rohy jsou jedním z kotevních bodů APR algoritmu. Tyto body jsou vytvořeny z největších špiček *x* a *y*-histogramů, které zobrazují úhlový rozdíl mezi lokalizací robotu a úhlem svírajícím s okolními objekty.

Skokové rohy jsou takové body, které mají zřetelnou pouze jednu hranu rohu. Jsou nejsnáze detekovatelné, a tyto rohové body jsou přítomny na hranicích dvou kolmých ploch při skenování těchto ploch laserovým paprskem. Při tomto skenování se bod objeví na bližší ploše.

Úhlové rohy jsou rohy, které mají viditelné obě strany. Jsou viditelné při spojitém přechodu mezi dvěma plochami. Úhlový a skokový roh se dá zaměnit návzájem v závislosti na tom, jak se na snímaný roh díváme.

3.4 Využití pomocí histogramu

Tato poslední metoda, kterou se v práci budu zabývat je dalším přijatelným řešením korelace snímků a jejich zarovnání. Korelaci můžeme použít pouze na skeny, které se navzájem od sebe



Obr. 6 Metoda lokace pomocí histogramu. [24]

odlišují v rotaci. Pokud se skeny odlišují rotací a translací navzájem, rozložení úhlů a následný výpočet korelační funkce může vyjít s chybnými výsledky a z něj pak mohou být vyvozeny špatné závěry algoritmu. Z naměřených úhlů je tedy nutné vytvořit úhlový histogram. Pomocí histogramu můžeme poté vyjádřit nejvíce svírané úhly mezi senzorem a okolím. Totožný postup je poté prováděn u aktuálního skenu. [12]

3.5 Algoritmus založený na vzájemné korelaci

V této bakalářské práci budu navazovat na metodu, která byla navržena a popsána v literatuře [5]. Autor se věnoval implementaci algoritmu jako prototypu v jazyce `C#`. Tato práce bude popisovat nasazení algoritmu v jazyce `C`, která je vhodnější pro práci s mikrokontrolery.

Počátečním bodem praktické části této práce bylo opřít se o existující metody lokačního algoritmu, které jsou popsány výše. Tento algoritmus na předchozích vědeckých pracích staví a využívá jej jako základnu. Jádrem nové metody je využití korelačních způsobů k určení posunu a rotace nasnímaných laserových snímků. Pokud se některá změní, změna působí korelativně na druhou a naopak. Pokud je tedy přítomná korelace mezi dvěma snímky, můžeme s velkou jistotou říct, že na sobě závisí. [13] Laserový snímek snímaný senzorem nám ukazuje prakticky půdorys místnosti, kde se senzor nachází a snímá. Lidar obsahuje množinu bodů pro daný úhel[3] a v něm obsaženou informaci o vzdálenosti nejblíže umístěné překážky od snímače.

Uplatněme tedy korelační postup a nasnímejme dva různé snímky, kdy laserový snímač změní polohu. Můžeme s jistotou říci, že oba snímky budou podobné a navzájem se ovlivňující. Cílem je tedy najít afinní transformaci **m** takovou, která s co nejmenší odchylkou zobrazuje testovací sken na sken referenční. Afinní transformace na R^n je takové zobrazení A: $\mathrm{R}^n \to \mathrm{R}^n$, pro které existuje vektor $\mathbf{t} \in \mathrm{R}^n$ a lineární transformace $A : \mathrm{R}^n \to \mathrm{R}^n$ podle vzorce: [14]

$$A(\mathbf{x}) = A(\mathbf{x}) + \mathbf{t} \quad \forall \mathbf{x} \in \mathrm{R}^n \tag{2}$$

Práce navazuje na metodu, která používá k nalezení afinní transformace modifikovanou korelaci. Parametry **m** jsou posun a rotace.

$$\mathbf{m} = (m_x, m_{y,}\omega) \tag{3}$$

kde m_x je posun ve směru x, m_y je posun ve směru y a ω je rotace snímku.

Afinní transformaci můžeme vyjádřit takto:

$$\begin{pmatrix} i_x \\ i_y \end{pmatrix} = \begin{pmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{pmatrix} . \begin{pmatrix} j_x \\ j_y \end{pmatrix} + \begin{pmatrix} m_x \\ m_y \end{pmatrix}, \tag{4}$$

kde i_x, i_y a j_x, j_y jsou souřadnice jednoho bodu a ω je rotace lidaru.

Hledáním existujících metod zmíněné afinní transformace jsme věnovali Kapitolu 3.

3.5.1 Hledání afinní transformace

Snímky lišící se pouze rotací

V této kapitole se budeme zabývat vhodnou afinní transformací. Mějme dostatečně rozmanité prostředí tak, aby funkce $r(\Phi)$ a $s(\Phi)$ měly periodu T = 2π.

[3] použitý senzor SICK LMS 100 snímá s krokem 0,25 ° nebo 0,5 °

Korelační funkce poté definujeme jako:

$$(r * s)(\Phi) = \int_0^{2\pi} r(\tau) \cdot s(\Phi + \tau) \cdot d\tau \tag{5}$$

Výsledkem výše zmíněné funkce je ukázat míru zobrazení v závislosti na úhlu Φ. Pro požadovanou rotaci Φ_M platí:

$$(r * s)(\Phi_M) = \max\{(r * s)(\Phi)\} \tag{6}$$

Snímky lišící se rotací a zároveň posunem Jsou dány dva lidary A a B od sebe odlišné rotací a posunem, což je v praxi mnohem častější úkaz. Abychom získali informaci o kolik se senzor posunul a otočil, je třeba aplikovat modifikovanou korelaci. Takovou funkci můžeme popsat jako funkci tří parametrů $(r * s)(m_x, m_{y,}\omega)$. Pro získání posunu je tedy důležité vyselektované snímky převést do kartézských souřadnic. Funkce pro převod do kartézských souřadnic je popsána níže:

$$\begin{aligned} x &= r \cdot \cos(\varphi) \\ y &= r \cdot \sin(\varphi) \end{aligned} \tag{7}$$

kde x je x-ová souřadnice bodu v kartézské soustavě a

y je y-ová souřadnice bodu v kartézské soustavě a

φ je úhel svírající s osou x a

r je vzdálenost bodu od senzoru.

Modifikovanou korelační funkci definujme jako:

$$(r * s)(m_x, m_y, \omega) = \begin{pmatrix} r \cdot \cos(\varphi) \\ r \cdot \sin(\varphi) \end{pmatrix} \odot \begin{pmatrix} s(\varphi + \omega) \cdot \cos(\varphi + \omega) + m_x \\ s(\varphi + \omega) \cdot \sin(\varphi + \omega) + m_y \end{pmatrix} \tag{8}$$

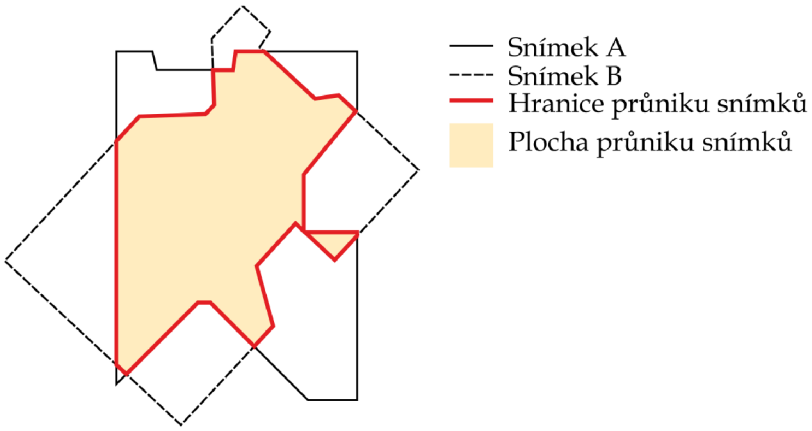
kde $(r * s)(m_x, m_y, \omega)$ je modifikovaná korelační funkce a $\otimes$ je nahrazená tradiční korelace, jejímž výsledkem je kladná hodnota, která znázorňuje velikost podobnosti referenčního a testovacího laserového snímku senzoru. Dalo by se říci, že tato část je esenciální částí navrženého algoritmu a určuje nejvhodnější afinní transformaci **m**. Pro transformaci $\mathbf{m} = (x_M, y_M, \omega_M)$ tedy platí:

$$(r * s)(x_M, y_M, \omega_M) = \max\{(r * s)(m_x, m_y, \omega)\} \tag{9}$$

Naším cílem je tedy najít pro danou korelaci maximální kladnou hodnotu, která pak značí maximální podobnost dvou laserových snímků. Korelační koeficient, který získáme pomocí

operace $\odot$, je suma násobků hodnot jednotlivých funkcí v daném místě. Pro každý výpočet je nutný jeden průchod, kdy spočítáme násobky obou funkcí, které následně sečteme. Tento průchod je nutný spočítat pro každé natočení. Výpočet korelačního koeficientu tedy vyžaduje dva vnořené průchody, kdy jeden průchod spočítá hodnotu koeficientu pro dané natočení a druhý průchod spočítá hodnotu koeficientu pro každé natočení. Pro nasazení algoritmu v praxi je také ale potřeba uvažovat v trojrozměrné rovině. K dvěma průchodům (posun a rotace) je třeba implementovat třetí vnořený průchod - osa z. Ohledně vnořených cyklů[4] je také třeba vzít v potaz časovou náročnost samotné modifikované korelační funkce. Nejdelší dobu trvání při zpracování algoritmu je tudíž tato korelační část.

Schéma 7 znázorňuje situaci dvou vzájemně natočených snímků. Způsob, jakým se můžeme dopracovat ke korelačnímu koeficientu, je vypočítat obsah překrývajících částí nasnímané místnosti. Maximální překrytí (plocha) tedy bude znamenat také maximální korelační koeficient. Ačkoliv se tato metoda může zdát jednoduchá, z hlediska výpočetní náročnosti je neuskutečnitelná, pokud budeme uvažovat o nasazení na mikrokontroler. Mějme výsledek průniku dvou snímků jako obecný n-úhelník, přičemž snímky uvažujme také jako obecné n-úhelníky. K výpočtu samotného tvaru průniku je možné použít Weiler-Athertonův algoritmus [15]. Lidary obsahují až tisíce hodnot, a výpočet průniku by byl časově náročný při myšleném použití mikrokontroleru. Dále je ještě potřeba vypočítat obsah takového n-úhelníku.

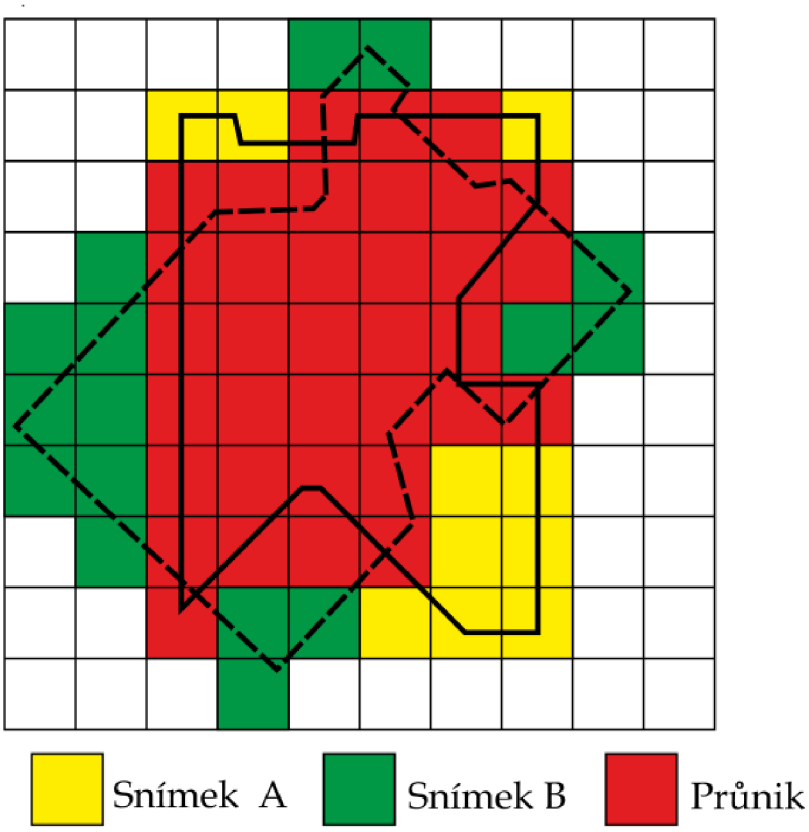


Obr. 7 Vzájemně natočené snímky. [25]

Další způsob, jak zjistit koeficient je znázorněn na obrázcích 8 a 9. Pro výpočet je nutné převést snímek na rastrový. Jedním ze způsobů je vyplnit obsah plochy, která je ohraničena n-úhelníkem. Tato metoda se v porovnání s druhou metodou, která bude popsána níže, ukázala jako výpočetně

[4] V praktické části bakalářské práci jsou využívány cykly *for*

náročnější. Je zde totiž nutnost zjistit, zda se bod nachází vně nebo uvnitř polygonu. Metoda je znázorněna na obrázku 8.

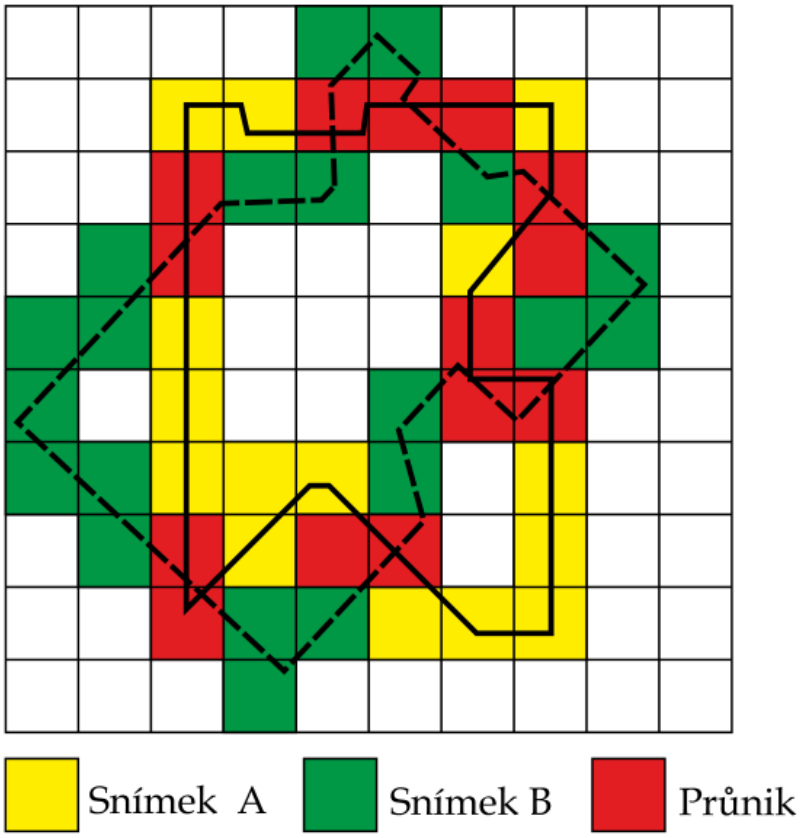


Obr. 8 Výpočet korelačního koeficientu - postup A. [25]

Odlehčená varianta, která je zaznamenána na obrázku 9 je vhodnější co se týče výpočetního času a náročnosti. Uvažujme tedy jen o hranicích daných n-úhelníků. Pokud se na rastrovém snímku v buňce nachází oba body hranic navzájem posunutých a otočených snímků, označí danou buňku. Navíc zde uvažujeme jen o buňkách, kde je alespoň jeden naměřený bod. Tudíž odpadá nutnost zpracovávat celou mapu, ale jde jen o vypracování určité části mapy rastru. Prerekvizitou pro tuto metodu je dostatečně husté bodové rozložení.

3.5.2 Výpočet korelačního koeficientu

V této podkapitole se budeme zabývat postupem výpočtu korelačního koeficientu z předchozího textu. Máme tedy referenční a testovací rastrovu mapu se zaznačeným půdorysem budovy.



Obr. 9 Výpočet korelačního koeficientu - postup B. [25]

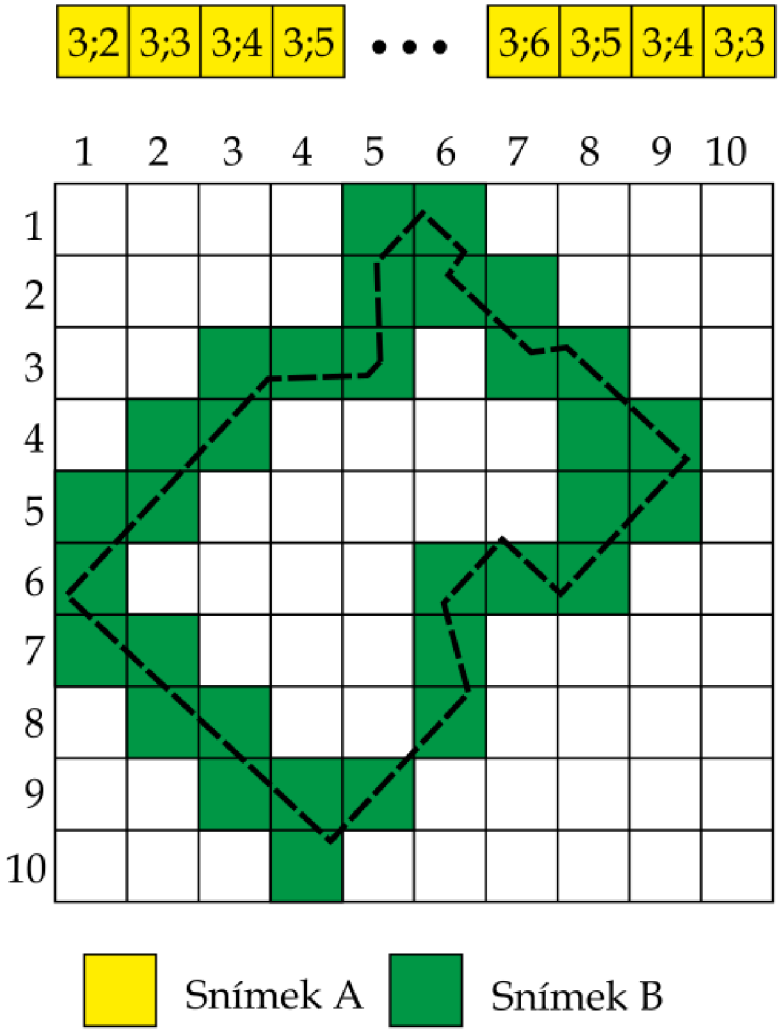
Pro zjištění míry podobnosti je třeba projít rastr referenčního snímku, a porovnat s rastrem testovacího snímku. Vezměme si pro názornost obrázek 9. Počet kroků, který je zapotřebí k prozkoumání celého rastru, je 100. Velikost mapy je totiž 10 na 10. Přičemž při samotném snímání bodů a ukládání do rastru byla vytvořena řídká matice, tzn. matice, která obsahuje pouze buňky, které mají minimálně jeden nasnímaný bod. Odkažme se tedy znovu na obrázek 9. Počet buněk, které obsahují miminálně jeden bod z nasnímaného půdorysu budovy je 29. Výpočetní náročnost je v tomto případě třikrát menší než při použití konvenčního procházení rastru, tzn. bod po bodu. Avšak je důležité, aby při zanášení bodů do rastru byla také zapsána informace ohledně pozice pro každou buňku. Řídká matice je znázorněna na obrázku 10.

Prvním krokem navrženého algoritmu je tedy vyzvednutí jedné buňky z množiny referenčního snímku. Dále pak ověřujeme, jestli se v testovacím snímku na daných souřadnicích nachází buňka rastrové mapy. Pokud se bod nachází na dané buňce, navýšíme hodnotu koeficientu o číslo 1. Tyto

kroky opakujeme pro každý bod z naměřené množiny bodů referenčního snímku. Poté porovnáme hodnoty koeficientu a vybereme hodnotu, která je maximální.

3.6 Prerekvizity algoritmů

Pro proveditelnost všech algoritmů je také potřeba zanalyzovat prerekvizity nutné pro výše popsané metody. Tyto prerekvizity jsou například nutné pro proveditelnost výpočtu korelačního koeficientu a v dalším případě nutné pro správné výsledky, které jsou nezkreslené, či v co



Obr. 10 Řídká matice. [25]

nejmenší míře zkreslené. Klíčovou podmínkou pro proveditelnost algoritmu je dostatečná hustota dat. Tímto se rozumí dostatečné množství bodů nasnímaných laserovým skenerem pro jakýkoliv snímek. Tuto kardinální překážku ve výpočtu bychom mohli simulovat tím, že umístíme senzor do otevřené nečlenité krajiny. Tímto nezajistíme senzoru dostatečný počet překážek pro vhodný počet bodů. Dostáváme se k další podmínce, a to tvar prostředí, ve kterém se senzor nachází. Tyto specifické tvary místnosti mohou zapříčinit špatné vyhodnocení translace a rotace navzdory dostatečnému počtu bodů.

Středově souměrná prostředí

Jedním z tvarů prostředí, které nesplňují prerekvizity algoritmu je středově souměrná místnost. Tento typ místnosti zásadně negativně ovlivňuje pravdivost výsledku. Může dojít k chybnému výpočtu otočení senzoru a tedy i k chybnému výpočtu posunu. Tento jev je zapříčiněn už v samotném jádru algoritmu a jedná se o počet lokálních maxim korelační funkce. Funkce má v tomto prostředí několik lokálních maxim.

Pravidelné tvary prostředí

Toto prostředí způsobí špatnou kvalitu zarovnání snímků. Jak již bylo zmíněno výše, korelační funkce tohoto prostředí má několik lokálních maxim. Díky tomu opět dojde k špatnému určení posunu.

Prostředí s opakujícími tvary

Tento typ prostředí můžeme popsat jako prostředí s více než jedním maximem. Tuto situaci můžeme také popsat jako opakující se tvar v x-ové nebo y-ové ose. Příklad takového prostředí je dlouhá chodba s pedestiály či sloupy. Tento tvar prostředí má opět negativní vlivy na správný výpočet lokačního algoritmu.

Kruhová místnost

V této místnosti, která je středově i osově souměrná nemá senzor žádné maximum pro vyhodnocení korelačního maxima. Proto tento typ prostředí negativně ovlivňuje funkcionalitu algoritmu.

Další pravidelné tvary

3.7 Závěr rešerše

V předchozí kapitole jsme se věnovali existujícím metodám pro zarovnání laserových snímků. Pro každou metodu jsou význačné jiné rysy a přednosti. První zmíněnou metodou byl ICP algoritmus, který se řadí mezi bodově orientované metody. Tyto metody se vyznačují větší početní zátěží, která vede k vyššímu časové náročnosti. Tento algoritmus vyhledává pár bodů,

které následně upraví eliminováním vzdáleností mezi nimi. Nevýhodou tohoto algoritmu je velká výpočetní náročnost, jelikož jsou výpočty iterační. Další nevýhodou tohoto algoritmu je nutnost do automatizovaného systému implementovat přídavné senzory a algoritmy pro eliminaci možného nesprávného vyhodnocení výsledku, například odometrie. Tato nutnost vede ke zvýšení ceny systému, náročnost na mikroprocesor a v neposlední řadě časovou náročnost. Tato nutnost je z důvodu možnosti, že iterační proces nebude konvergovat, ale naopak divergovat.

Další metodou je MCL (Monte Carlo lokalizace), která používá statistický model. Jeho základem je částicový filtr, který prezentuje pravděpodobnost stavu okolí, tedy šance výskytu robotu na určeném místě je váha bodů a jejich zastoupení na daném místě. Nevýhodou této metody je již zmíněná nutnost odometrie.

Poslední metodou byl algoritmus, jehož funkcionalita je postavena na základě korelačních funkcí. Podstatou této funkce je tedy porovnávání dvou laserových lidarů pomocí korelace[5] a následné vyhodnocování na základě korelačního koeficientu zjišťujeme v jaké rotaci a posunu se snímky sto procentně překrývají. Z důvodu využití laserového senzoru vzniká nevýhoda v tom, že senzor nedokáže detekovat skleněné překážky a průhledné předměty (plasty a podobně). V moderní době, kdy se budovy skládají z velké části ze skla je tato nevýhoda nežádoucí. Je třeba implementovat senzor fungující na jiné vlnové délce, který je schopen detekovat tyto průhledné překážky. Takový senzor, který dokáže detekovat sklo, je například přístroj firmy SICK, například SICK W12G [20]. Je nutné ale podotknout, že se tato nedostatečnost laserových senzorů vyskytuje i u MCL, ICP a dalších algoritmů založených na těchto senzorech. Nicméně metoda založená na cross korelaci nevyžaduje kromě snímače skla a průhledných materiálů žádný další přídavný senzor.

Z experimentálního měření, které bylo provedeno v této práci [5], můžeme reprodukovat výsledky a vyvodit z nich výhody a nevýhody. Výhodou této metody je, že dokáže vyhodnotit translaci a rotaci v poměrně proměnlivém prostředí, kdy se dynamicky pohybují různé objekty. Můžeme si tuto situaci představit jako lokační algoritmus v místnosti, kde se pohybuje velké množství lidí. Pokud pomineme nutnost detekovat skleněné a průhledné překážky, můžeme konstatovat, že není třeba žádných dalších přídavných senzorů pro přesné určení lokace. Tento senzor je také použitelný pro přístupy SLAM[6]. Experiment dokázal, že popisovaný algoritmus je pro tento přístup SLAM plně funkční. Mezi výhody můžeme také zařadit robustnost[7] algoritmu, která je u této metody dostatečně široká vůči prostředí.

[5] V programovacím jazyce Matlab je korelační funkce používána příkazem **xcorr**.

[6] SLAM - simultaneous localization and mapping

[7] Snaha o naprogramování algoritmu, který se při chybě (atypický jev) nezastaví, ale dokáže pracovat dál[27]

4 Realizace algoritmu v praxi

V této části bakalářské práce se budeme zabývat výsledky algoritmu v praxi a následným ověřením navrhované metody. Pro tuto část práce byl sepsán program v jazyce C[8]. Tento program se připojuje přes socket[9] server, který zajišťuje komunikaci, na laserový senzor SICK LMS 100 [16]. Senzor vysílá data Wi-Fi signálem přes router, kterým se připojíme programem. Tento senzor je na obrázku 11.

Obr. 11 Laserový senzor SICK LMS 100. [26]

4.1 Laserový senzor pro navržený algoritmus

V této podkapitole se budeme věnovat použitému laserovému senzoru. Pro pochopení algoritmu je potřeba se nejprve zabývat daným přístrojem, který odvede svou část práce, a to vrácení informací o jednotlivých bodech. Použitý senzor se řadí mezi laserové senzory, které se v dnešní době těší velké oblibě nejen na poli robotiky a automatizace. Vědeckých článků a literatury ohledně laserových senzorů bylo sepsáno už mnoho. [17, 18, 19]

[8] Počátkem 70. let 20. století vyvinuli programátoři Thompson K. a Ritchie D. jazyk C pro OS Unix. Jazyk *C* je jednoduchý, nenáročný a široce oblíbený, zvláště pro nasazení na méně výkonné čipy.

[9] Síťový socket je koncový bod dané komunikace v počítačové síti.

Princip senzoru

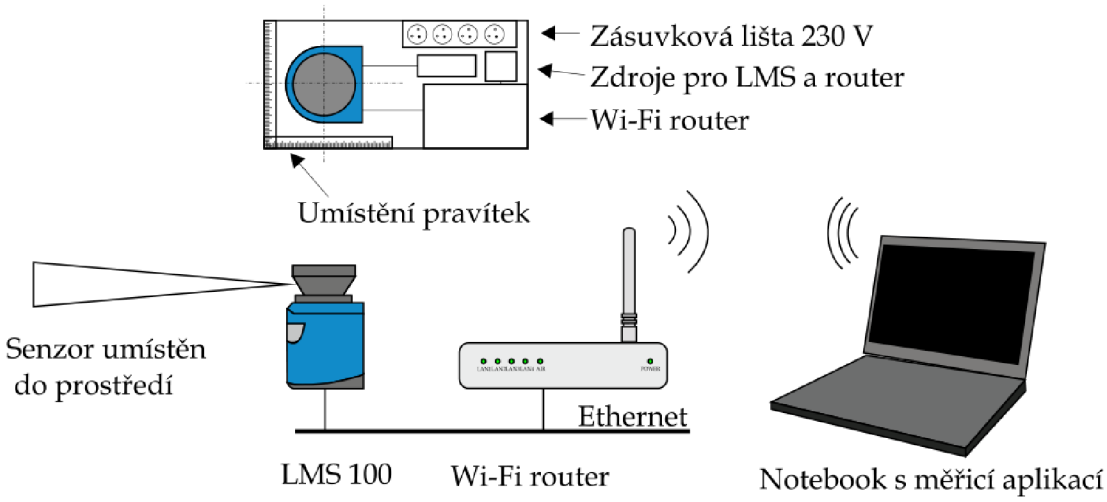
Primární funkcí senzoru kvantového generátoru světla je výpočet zpětně reflektujícího proudu fotonů koherentního svazku. LMS 100 obsahuje motorek, který posouvá svazek od jednoho maxima po druhé maximum senzoru. Díky tomuto získáváme informace o bodech s pevným úhlovým odstupem mezi sebou. Tento úhlový odstup činí většinou 0,25° nebo 0,5°. Zhotovíme si jakýsi laserový řez okolí. Tyto body pak následně používáme pro mnoho výpočtů, od určování translace, rotace a tvaru prostředí až po implementaci SLAM a jeho tvorbu mapy. Jak již bylo zmíněno výše, senzor SICK LMS 100 pracuje na bázi laserového zdroje. Vlnová délka laseru u tohoto senzoru se pohybuje v pásmu infračerveném s vlnovou délkou 905 nm. Úhel snímání senzoru je 270°, kdy vrací zpět údaje o jednotlivých bodech objektů v okolí. Takových řezů okolím dělá senzor zhruba 25 nebo 50 za sekundu. Rozsah senzoru, kdy dokáže nasnímat nejvzdálenější bod, je 20 metrů. Senzor je limitován nejkratší vzdáleností od překážky, a to 0,5 metrů. Co se týče rozhraní, můžeme použít RS-232[10], Ethernet[11] nebo CAN[12]. Pro co nejvěrnější simulaci budoucího použití algoritmu byla zvolena kombinace ethernetového kabelu s Wi-Fi routerem. Takto bude zajištěna bezdrátová komunikace mezi senzorem a počítačem.

Připojení laserové senzoru k wifi routeru a napájení samotného senzoru je zobrazeno na obrázku 12.

[10]Nebo-li sériová linka používána nejčastěji jako komunikační rozhraní.

[11]Kabely využívající čtyři páry kroucené dvojlinky s modulárním konektorem 8P8C.

[12]Controller Area Network (CAN) je nejčastěji využíván jako komunikační síť senzorů. Je hojně používána v automobilové technice, odkud se pak rozšířila do průmyslové automatizace. Jde o sériovou datovou sběrnici vyvinutou formou Bosch GmbH



Obr. 12 Znázornění procesu vysílání dat senzorem. [25]

4.2 Implementace algoritmu v jazyce C

V této kapitole bude popisován postup z praktické části. Bude tedy popisován postup řešení a implementace algoritmu v jazyce C, který byl kompilován v programu Microsoft Visual Studio[13]. Styčné body postupu byly nadefinovány:

1. Příprava
2. Implementace
   (a) Vytvoření socket–server
   (b) Příjem a zpracování datagramu
   (c) Tvorba grid mapy a vsazení bodů
   (d) Výpočet translace a rotace
3. Testování

Příprava

Na počátku práce bylo třeba si nastudovat základy fungování laserových skenerů a již existující způsoby. Další studium se týkalo implementovaného algoritmu. Po teoretické přípravě bylo krátké období, kdy jsem si prakticky zopakoval základní syntax programovacího jazyka *C*.

V první řadě praktické části bylo předpřipravit si základní matematické funkce jako jsou základní goniometrické funkce, převod na základní úhel a podobně. Tabulku nadefinovaných funkcí si můžeme prohlídnout níže.

Funkce	**Název**	**Matematické vyjádření**
Sinus	sin_function	$\sin(\alpha \cdot \frac{\pi}{180})$
Cosinus	cos_function	$\cos(\alpha \cdot \frac{\pi}{180})$
Absolutní hodnota	abs_value	$\lvert a\rvert = \begin{cases} a, \text{pokud } a \geq 0 \\ -a, \text{pokud } a < 0 \end{cases}$
Základní úhel	BasicAngle	$\alpha = \alpha + 360, \text{pokud } \alpha \leq 0$ $\alpha = \alpha - 360, \text{pokud } \alpha \geq 0$

Tab. 1 Nadefinované funkce

[13] Kompilátor společnosti Microsoft, který překládá kód do binární podoby pro procesor.

Dále bylo potřeba si v rámci přípravy připravit funkci pro převod úhlů ze stupňů na radiány. Hodnota úhlového kroku v radiánech je dále používána při krokovém rozdílu výpočtu korelace při počítání rotace. K tomuto kroku se dále pak zpětně odkážu v dalším textu. Dalším krokem bylo nadefinovat si konstanty a proměnné a přiřadit jim fixní hodnotu. Tento soubor jsem si uložil do knihovny `defines.h`[14]. Tato knihovna obsahuje nadefinované proměnné, ale také i struktury, které byly přidávány při práci na algoritmu.

Funkce, kterou je také potřeba nadefinovat je výpočet souřadnic v kartézském tvaru. Tato funkce bude mít důležitou úlohu při sestavování 2D mapy a následné zakreslení bodů do mapy. Pro toto zakreslení bude potřeba znát x-ové a y-ové souřadnice bodů. Níže ukázka kódu znázorňuje jádro funkce pro přepočet kartézských souřadnic, pokud známe vzdálenost bodu od senzoru a úhel, pod kterým svírá s kladnou částí osy x.

```
for (i = 0; i < SCAN_SIZE; i++)
{
  scan->x[i] = scan->raw_data[i] * cos(angle);
  scan->y[i] = scan->raw_data[i] * sin(angle);
}
```

Tímto příprava algoritmu na implementaci končí.

Implementace

V první řadě je potřeba načíst data posílané laserovým senzorem SICK LMS 100. Toto hardwarové řešení senzoru bylo již vyřešeno autorem tohoto algoritmu. Jak již bylo zmíněno výše, senzor vysílá data přes wifi router, na který se následně připojujeme. Router vysílá pod jménem D400 a připojujeme se na něj klasicky jako na jakoukoliv Wi-Fi síť. Podle dokumentace [21] senzor posílá data v hexadecimální[15] formě. Samotná data jsou ale z obou stran ohraničena informačními údaji o senzoru jako je teplota okolí, vlhkost okolí nebo status senzoru. K připojení na senzor, na kterém běží server se připojujeme pomocí socket clienta. K tomu je potřeba includovat knihovnu winsock2.h[16], která má předdefinované příkazy nutné k správnému přijímání a odesílání dat ze sensoru. Port[17] je *2112* s IP adresou[18] *192.168.0.5*. Pokud máme vyřešenou konektivitu se senzorem, je třeba vyřešit problém se samotným zpracováním dat.

[14] Koncovka .h značí knihovnu.

[15] Číselná soustava základu 16.

[16] Tato knihovna se liší podle operačního systému počítače ze kterého se připojujeme na daný router. Winsock2 se používá pro Windows OS, sys/socket.h pro Linux OS.

[17] Číslo kanálu, přes který probíhá datový tok pomocí protokolů TCP a UDP. Tyto protokoly slouží k rozlišení aplikace.

[18] Číslo, které jednoznačně identifikuje síťové rozhraní v síti.

Nejprve bylo třeba si nastudovat dokumentaci [21] a zjistit jak aktivovat senzor, aby začal snímat a posílat data. Příkazem send_data s parametrem *sRN STlms* nejprve zjistíme stav senzoru. Pokud je vše v pořádku, pokračujeme stejným příkazem, avšak s parametrem*sEN LMDscandata 1*. V tomto okamžiku začneme přijímat data ze senzoru, avšak je nutné zaimplementovat zahazování prvních datagramů, jelikož senzor posílá datagramy ještě dřív, než započne jeho samotné měření. Tělo datagramu těchto snímků obsahuje jen nuly. Po zahození prázdných zpráv začneme přijímat validní data ve formátu:

```
[STX]sSN LMDscandata 1 1 9EFF58 0 0 8F4 1AE2
1334E5E9 1334F7E5 0 0 7 0 0 9C4 168 0 1
DIST1 3F800000 00000000 FFF92230
9C4 439 1D45 1D28 1D22
.
.
DATA
DATA
DATA
.
.
147 131 146 14C 14C 0 0 0 0 0 0[ETX]
```

Datová zpráva výše je jen ukázka. Jak si můžeme všimnout, začátek zprávy je označen netisknutelným znakem [STX], který reprezentuje v ASCII tabulce 0x02. Konec zprávy je naopak označen netisknutelným znakem [ETX], který reprezentuje znak 0x03. Datový obsah zprávy, který obsahuje vrácené vzdálenosti jednotlivých bodů můžeme oddělit od začátku zprávy. Jak již bylo zmíněno výše, začátek zprávy obsahuje informační údaje o prostředí a o samotném senzoru. Užitečná data pro korelaci můžeme oddělit od začátku datového informačního balíku tím, že oddělíme přesně 27 mezer od netisknutelného znaku [STX]. Poté hodnoty na pozici po 27. mezerníku až do [ETX] považujme za validní data. Jak bylo popsáno výše, je třeba ošetřit situace, kdy začneme přijímat zprávu, která nezačíná [STX]. Je proto třeba navrhnout systém spojování textových řetězců k sobě. Pro tuto funkci byl použit příkaz memcpy, který má v sobě tři parametry. Na prvním místě je místo, kam spojujeme dotyčný string[19], na druhém místě je dotyčný string a na třetím místě jakou má string velikost. Takto ošetříme situaci, kdy příjmeme neúplnou zprávu, které konec ještě zpracujeme a nadefinovaná funkce bude spojovat stringy dohromady a rovnou je analyzovat. Tímto se vyhneme tomu, že budeme zahazovat nějaký řetězec

[19]Datový typ řetězce.

a tím pádem docílíme snížení časové náročnosti. Jak již bylo zmíněno výše, dostatečná hustota dat je jednou z prerekvizit algoritmu. Zahazováním celého datového souboru, respektivě zbytku datového souboru, bychom přišli o důležité hodnoty.

Pokud máme zpracovanou zprávu, můžeme pokročit k dalšímu kroku, což je překlad z hexadecimální soustavy na desítkovou soustavu. Po naimplmentování převodníku (viz. ukázky kódů) máme tedy 1082 bodů s určenými kartézskými souřadnicemi. V této chvíli je potřeba vytvořit grid map, do které vložíme jednotlivé body. Mapu si vytvoříme tímto způsobem:

- Nadefinujeme si mezní zápornou hodnotu na ose x = -10 000 mm.
- Nadefinujeme si mezní kladnou hodnotu na ose x = 10 000 mm.
- Nadefinujeme si velikost jednoho čtvercového dílku = 200 $\times$ 200 mm.
- Osadíme každou buňku mapy hodnotou 0.

Výsledná mapa má tedy rozměry 20 $\times$ 20 metrů. Teď už nám nic nebrání osadit mapu body, které jsme získali ze senzoru. Ke každému bodu máme přidělené kartézské souřadnice. Získáváme tedy čtverečkovanou mapu s označenými dílky, které v sobě drží minimálně jeden bod. Z experimentálního měření bylo zjištěno, že je třeba zahodit body, které mají hodnotu souřadnic menší néž 200 mm (senzor neumí správně měřit moc blízké překážky). Pokud senzor z nějakého důvodu nedokázal změřit hodnotu daného bodu, vracel hodnotu 0. Tato chyba má pak negativní následek na korelační funkci, která vyúsťuje v nepřesnost zobrazení a výpočtu korelace. Také je třeba zahodit body, které přesahují velikost mapy. Pro co nejmenší časovou nenáročnost algoritmu je třeba implementovat při osazování bodů do mapy funkci, která si bude rovnou ukládat souřadnice dílků, kam připadne jednotlivý bod. Při sázení bodů do grid mapy si rovnou ukládáme souřadnice jednotlivých dílků kam jednotlivý bod spadá. Tento důležitý krok pak uplatníme při korelační funkci. Úspora je principiálně v tom, že namísto procházení celé mapy, která má 10 000 na 10 000 dílků budeme procházet jen místa, kde jsou nějaké body. Takto dochází k výrazné úspoře času, jelikož se počet kroků zmenší řádově v desítkách procent.

Tímto si vytvoříme snímek, který nazveme referenční. Po tomto snímku následuje testovací snímek, který budeme porovnávat s referenčním. V Kapitole 3.5 bylo zmíněno, že korelační funkce se aplikuje na dva snímky, které jsou navzájem posunuté a otočené. Testovací snímek tedy budeme porovnávat se snímkem referenčním tím, že ho budeme posunovat a otáčet. Při každém kroku se vypočítá korelační koeficient a pro daný koeficient si uložíme hodnotu rotace a translace. Maximální hodnota korelačního koeficientu poté značí, že se oba snímky maximálně překrývají. Tímto získáme hodnotu posunu a otočení od původní pozice, čili předchozího skenu.

Tento postup se opakuje pro každý snímek. Každý testovací snímek je v dalším kroku snímkem referenčním. Při zjišťování rotace byl použit systém hrubé korelace následovaný jemnou korelací. Hrubá korelace spočívá v přibližném zjištění rotace, jelikož je úhlový krok $20\,^{\circ}$. Při zjištění přibližné hodnoty rotace nastupuje jemná korelace, kdy se v radiusu $\pm\,10^{\circ}$ uplatňuje úhlový krok $2\,^{\circ}$. Tímto docílíme další časové úspory a zároveň přesnosti v porovnání s tím, kdybychom celý rozsah analyzovali dvěma stupni nebo dvaceti.

5 Experimentální část práce

Příprava a měření

Tato část práce se bude věnovat experimentální části měření. Měření bylo prováděno v prostorách budovy Fakulty elektrotechniky a informatiky. Algoritmus byl spouštěn v *Release*[20] modu pro dosažení co nejobjektivnějších výsledků. Specifikace modelu notebooku na kterém bylo provedeno měření je znázorněno v tabulce 2.

Značka	DELL Inc.
Model	15z (I15Z-4801SLV)
Název procesoru	Intel Core i7-3537U
Rychlost procesoru	2,0 GHz
RAM	8 GB
Operační systém	Microsoft Windows 10

Tab. 2 Specifikace notebooku

Cílem měření bylo vyhodnotit čas, jak dlouho trvá samotná korelace, tudíž nás nezajímá čas, kdy se senzor připojí na socket a začne přijímat data. Pro účely měření času byla potřeba připojit knihovnu *time.h*. S touto knihovnou získáváme datový typ *clock_t*. Použití je znázorněno níže.

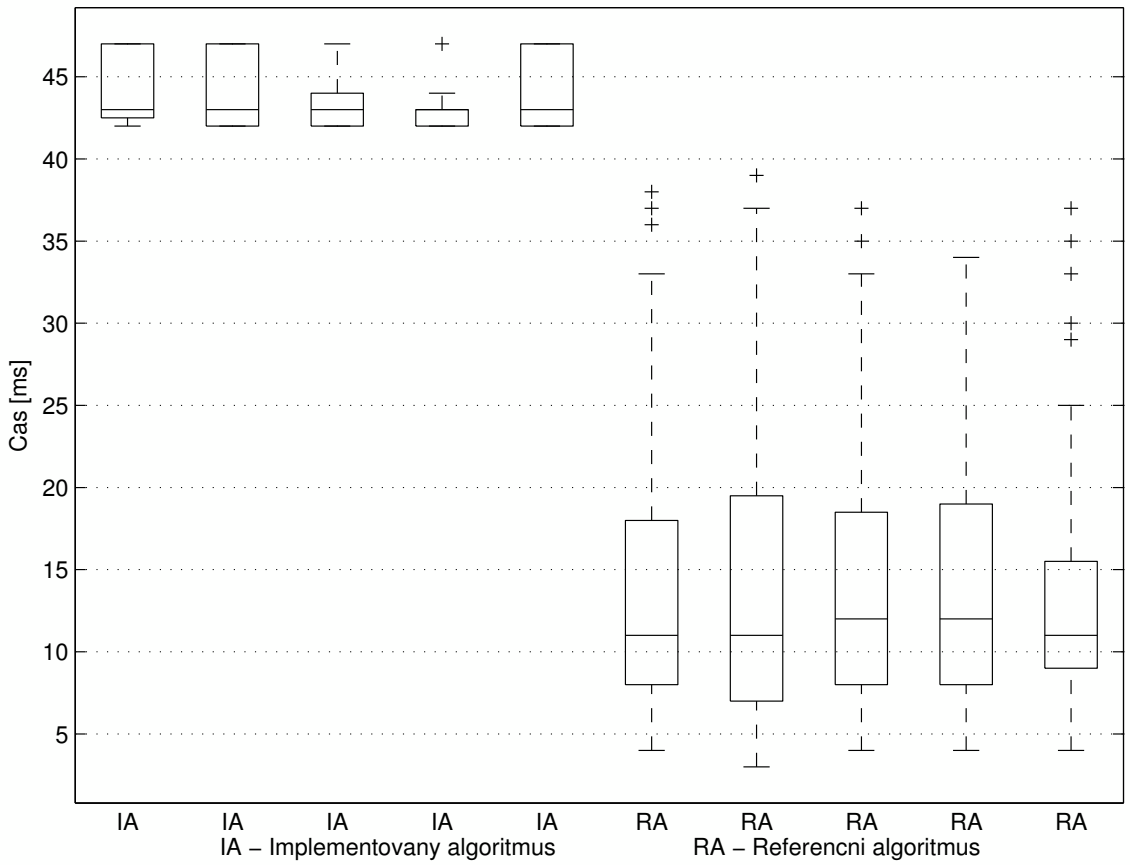
```
clock_t start = clock();
.
.
//korelace
.
.
clock_t end = clock();
```

Měření bylo uskutečněno na implementovaném algoritmu v jazyce *C* a na referenčním algoritmu, viz. kapitola 3.5. Samotných měření bylo pět, přičemž se při každém měření stokrát zaznamenávala délka, jak dlouho trvá korelace. Naměřené hodnoty se zapsaly do programu MATLAB[21] a přes funkci *boxplot()* se vykreslily krabicové grafy.

Vyhodnocení

[20] V release modu neobsahuje .exe soubor informace o debugování. Proto je CPU méně zatíženo.

[21] Scriptovací jazyk a kompilator pro matematické výpočty, vizualizace a analýzu dat.



Obr. 13 Krabicové grafy referenčního a implementovaného algoritmu

V grafu je obsaženo deset krabicových grafů, pro každé měření (100 hodnot) byl tedy vykreslen graf. Popisek na x-ové ose značí, jestli jde o implementovaný algoritmus (IA) nebo referenční algoritmus (RA). Na y-ové ose jsou hodnoty času v jednotkách milisekund.

Jak je vidět na tabulce 3, referenční algoritmus vychází časově méně než algoritmus implementovaný. Na první pohled se může zdát, že jsme tímto nesplnili hlavní motivaci této bakalářské práce, ale opak je pravdou. Důvodem proč čas nutný k tomu, aby se provedla korelace u referenčního algoritmu, je kratší než implementovaný je, že program algoritmu je psaný v C#, který dokáže pracovat ve vláknech[22]. Autor algoritmu nasadil dvě vlákna, tudíž výpočetní doba by se měla vynásobit dvěma. Tento čas ale stále není úplně pravdivý. Dalším krokem, který implementoval autor pro optimalizaci bylo vynechání korelace nahrubo. Jak již bylo zmíněno v

[22]Paralelní procesy, které vykonávají určité úkony. Můžeme tedy spouštět několik částí kódu v jednom okamžiku. Tímto se optimalizuje výpočetní doba.

Číslo měření	Referenční algoritmus [ms]	Implementovaný algoritmus [ms]
1	14,23	44,62
2	14,06	43,95
3	14,04	43,67
4	14,15	44,77
5	13,05	44,28

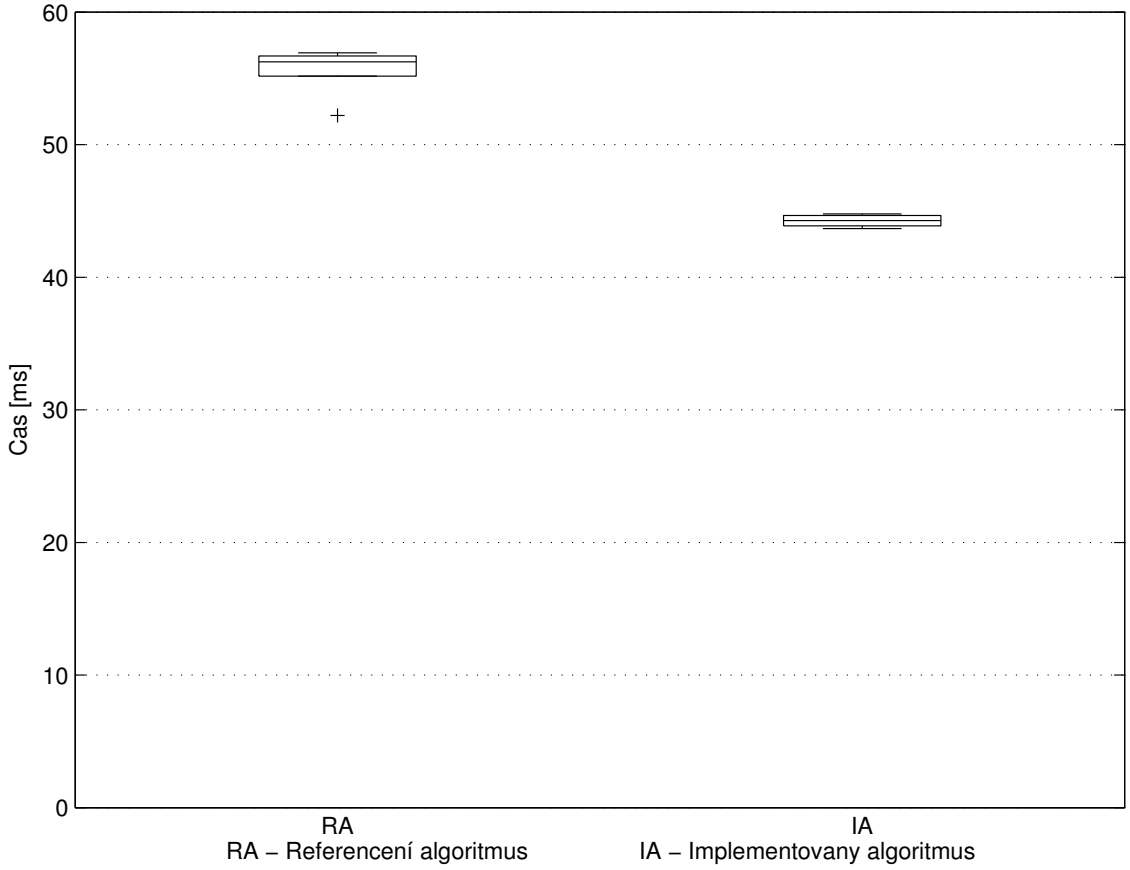
Tab. 3 Průměrné časy korelace

kapitole 3.5, hrubá korelace je úkon, kdy se prochází úhlový rozsah s úhlovým krokem $20\,^{\circ}$. Takto získáme přibližnou hodnotu rotace, která se zpracovává v jemné korelaci. Algoritmus, ze kterého tato práce vychází, tedy pracuje s předpokladem, že se senzor neotočí v krátkém čase 100 ms o více než $10\,^{\circ}$. Výsledný čas po zohlednění výše zmíněních poznatků je uveden v tabulce 4.

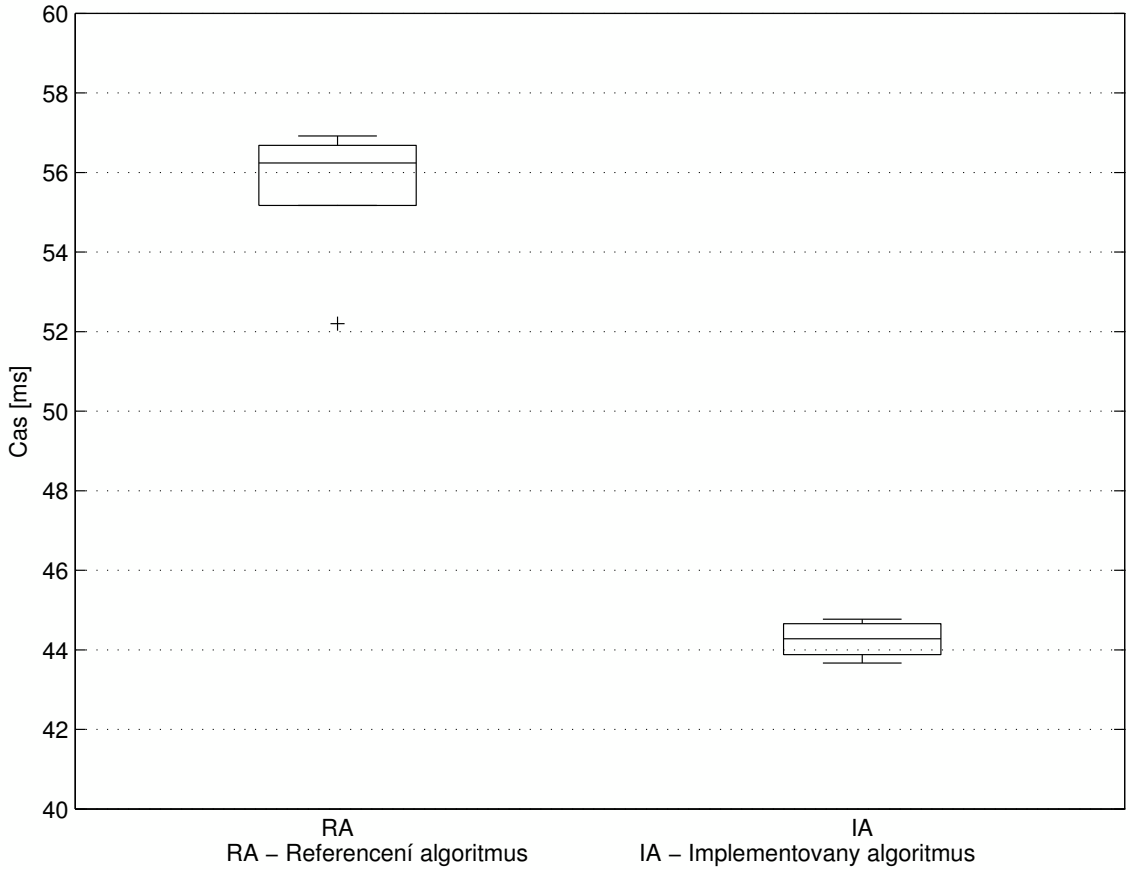
Číslo měření	Referenční algoritmus [ms]	Implementovaný algoritmus [ms]
1	56,92	44,62
2	56,24	43,95
3	56,16	43,67
4	56,6	44,77
5	52,2	44,28

Tab. 4 Průměrné časy korelace

Na obrázku 14 je znázorněn boxplot časů obou verzí algoritmu. Obrázek 15 je pak přiblížený graf pro lepší viditelnost. IA značí implementovaný algoritmus a RA značí referenční algoritmus. Hodnoty byly vzaty z tabulky 4. Pro každou verzi se spočítal aritmetický průměr a jeho hodnota se poté nanesla do grafu. Výpočtem bylo zjištěno, že implementovaný algoritmus má časovou úsporu o 20,35 %, čímž jsme splnili základní cíl této práce. Hodnoty se můžou měnit v závislosti na výkonnosti přístroje, na kterém bylo prováděno měření a také odchylkami při výpočtu.



Obr. 14 Krabicové grafy referenčního a implementovaného algoritmu po úpravě



Obr. 15 Krabicové grafy referenčního a implementovaného algoritmu po úpravě s menším rozsahem y-ové osy

6 Závěr

Tato bakalářská práce se zabývala implementací lokačního algoritmu založeného na vzájemné korelaci do programovacího jazyka *C*. V Kapitole 3 byly nastudovány metody již existující, které využívají laserové zarovnávání pro korelační funkci. Po nastudování teoretických znalostí, kde byly zhodnoceny klady a zápory, byl detailněji podroben studiu algoritmus založený na vzájemné korelaci. Bylo velice důležité, aby byl tento algoritmus pochopen a detailně prostudován pro následnou praktickou část.

Pro praktickou část v Kapitole 4 bylo potřeba zopakovat si základy programování v jazyce *C*. Tato praktická část byla důkladně zopakována naprogramováním výpočtu korelační funkce v rámci jedné roviny. Samotná implementace se skládala z několika podčástí. Prvním krokem bylo připravení matematických a geometrických funkcí pro správný výpočet požadovaných operací. Další částí byl výpočet translace a rotace dvou lišících se snímků. Testování a funkčnost byla prováděna na testovacích hodnotách. Po zprovoznění jádra algoritmu bylo potřeba vyřešit softwarově připojení se na senzor. Toto bylo vyřešenou naprogramování systému socket server. Další důležitou částí bylo zpracování příchozích dat, které byly dodávany velkou rychlostí ze senzoru. Tato data byla vysílána neustále a kontinuálně senzorem, proto program musel ošetřit, kdy byla zachycena neúplná zpráva. Zahazováním celého neúplného datagramu by zvýšila časovou náročnost algoritmu, proto tato možnost nepřipadala v úvahu. Datagram ze senzoru je v šestnáctkové soustavě, proto bylo potřeba nadefinovat vlastní překládač z šestnáctkové soustavy na soustavu desítkovou. Hotový a úplný diagram v desítkové soustavě je poté poslán dále pro korelační část programu. Poslední částí praktické části bylo ladění kódu programu, zvyšování efektivity a zkrácení výpočetního času. Zkrácení výpočetního času bylo například dosaženo rozdělením korelace na hrubou a jemnou korelaci. Hrubá korelace je, když se celý úhlový rozsah prochází s velkým krokem. Dostaneme tedy přibližnou hodnotu rotace senzoru. Pro zpřesnění výsledků je nasazena jemná korelace, která v rozmezí $\pm\ 10^{\circ}$ prochází daný rozsah.

Po úspěšném laboratorním testování odlazeného algoritmu nastala část experimentální. Tato část se skládala z praktické části a z části statistické, kde byly vyhodnocovány výsledky. Měření bylo prováděno v reálném prostředí za normálních podmínek. Cílem měření bylo ověřit správnost výpočtu posunu senzoru a rotace senzoru. Tento cíl byl ověřen následným manuálním měřením vzdálenosti a rotace. Dalším cílem bylo změřit dobu trvání výpočtu dané korelace s porovnáním referenčního algoritmu. Tato data pak byla zpracována programy MATLAB a MS EXCEL. Časy jednotlivých měření byly vykresleny krabicovými grafy. Pro zobjektivnění výsledků bylo potřeba upravit dobu trvání výpočtu u referenčního algoritmu tak, aby podmínkami odpovídala implementovanému algoritmu. Bylo potřeba zohlednit fakt, že

referenční algoritmus pracuje ve vláknech a také, že pracuje s předpokladem, že se senzor v čase 100 ms neotočí o více jak $10°$. Po zohlednění těchto faktorů, byl splněn i druhý cíl této práce – časová úspora oproti referenčímu algoritmu.

Dalším směrem, kam práce může ubírat je nasazení této implementace na mikrokontrolery. Důvodem této motivace je nízká cena součástek na nynějším trhu. Cenová dostupnost pak otevírá mnoho dalších možností a využití tohoto algoritmu v civilním i státním využití. Možnost práce na algoritmu je také na optimalizaci a snížení časové náročnosti programu. Minimalizace náročnosti pak opět může vést ke snížení ceny automatizovaného robotu, který by měl lokační algoritmus naimplementovaný.

Použitá literatura

[1] KJER, H., WILM, J. Evaluation of surface registration algorithms for PET motion correction. *In Bachelor thesis,* Technical University of Denmark, Informatics and Mathematical Modeling, Denmark, 2010.

[2] BESL, P., MCKAY, N. A Method for Registration of 3-D Shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (Los Alamitos, CA, USA: IEEE Computer Society),* Vol.17, No.8, pp. 239-256, 1992

[3] CHEN, Y., MEDIONI G. Object modelling by registration of multiple range images. Image Vision Comput. *Newton, MA, USA: Butterworth-Heinemann,* 145–155. doi:10.1016/0262-8856(92)90066-C.

[4] KJER, H., WILM, J. Evaluation of surface registration algorithms for PET motion correction. *Bachelor thesis,* Technical University of Denmark, Informatics and Mathematical Modeling, Denmark, 2010.

[5] KONEČNÝ, J. Principles of mobile service robots. *Dissertation thesis,* Vysoká škola báňská - Technická univerzita Ostrava (2014).

[6] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *In Proceedings of the Sixteenth National,* Conference on Artificial Intelligence. Orlando, USA, 1999.

[7] FOX, D., WOLFRAM, B., DELLAERT, F., THURN, S. Robust Monte Carlo Localization for Mobile Robots. Computer Science Department. University of Freiburg, Germany, 2001.

[8] CHEEIN, F., TOIBERO, J., DI SCIASCIO, F., CARELLI, R., LOBO PEREIRA, F. Monte carlo uncertainty maps-based for mobile robot autonomous slam navigation. *In Industrial Technology (ICIT),* 2010 IEEE International Conference on, pp. 1433–1438 (2010).

[9] THRUN S., BURGARD W., FOX D. Probalistic Robotics. *MIT Press,* 2005. Ch. 8.3 ISBN 9780262201629.

[10] ANTECKÝ, J. Aplikace metody zarovnání snímku pro lokalizaci autonomního robotu. *Master's thesis,* Vysoká škola bánská - Technická univerzita Ostrava (2014).

[11] WEBER, J., JÖRG, K., PUTTKAMER, E. APR - Global scan matcing using anchor point relationships. *In Proc. of the 6th international conference on intelligent autonomous systems,* IOS press, pp. 471-478, 2000.

[12] QUI, Q., HAN, J. Laser Scan Matching Using Multiplex Histograms with Feature Components. *In Proc.Int. Conf. on Robotics and Biometics,* Guilin, China, December 19-23. 2009.

[13] ZVÁROVÁ J. Biomedicínská statistika I. : Základy statistiky pro biomedicínské obory *Dotisk 1. vydání Praha : Karolinum, 1998. 218 s. ISBN 80-7184-786-0.*

[14] OLŠÁK, P. Afinní transformace [online] *2013 [cit. 2015-09-13]. Dostupné: http://petr.olsak.net/bilin/afinita-v2.pdf*

[15] WEILER, K., Atherton, P. Hidden Surface Removal using Polygon Area Sorting *Computer Graphics, 11(2):214-222, 1977.*

[16] SICK AG. LMS SICK 100: Laserový senzor [online]. [cit. 2015-09-26]. Dostupné z: *https://www.sick.com/de/en/detection-and-ranging-solutions/2d-laser-scanners/lms1xx/lms100-10000/p/p109841*

[17] DONG-GI W., JONG-KYU O., CHAN-HO L., SANG-HUN L., SUNG-HYUN J. Development of a multi-line laser sensor based robotic 3D measurement system *Control, Automation and Systems (ICCAS), 2011 11th International Conference on, 2093-7121, 2011.*

[18] KOVACS, E. a A.S.V. MS., Laser sensor based measurement system with 2D motion control. *International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2006. SPEEDAM 2006 [online]. 2006 [cit. 2015-09-26]. DOI: 10.1109/speedam.2006.1649854.*

[19] SAITO R., WATANABE K., NAGAI I., Laser odometry taking account of the tilt on the laser sensor. In: *2015 10th Asian Control Conference (ASCC) [online]. 2015 [cit. 2015-09-26]. DOI: 10.1109/ascc.2015.7244648.*

[20] SICK AG. Sensors for the detection of transparent objects. *Sensors for the detection of transparent objects [online]. [cit. 2015-10-03]. Dostupné z: https://www.sick.com/media/pdf/2/72/772/IM0047772.PDF*

[21] SICK AG. LMS SICK 100: Laserový senzor [online]. [cit. 2015-09-26]. Dostupné z: *http://www.hizook.com/files/publications/SICK_LMS100.pdf*

Citace obrázků

[22] Iterative Closest Point Algorithm [online]. F. Colas, 2011 [cit. 2015-11-30]. Dostupné z: *http://www.asl.ethz.ch/education/master/info-process-rob/ICP.pdf*

[23] WEBER, Joachim, Klaus-Werner JÖRG a Ewald von PUTTKAMER. APR - Global scan matcing using anchor point relationships [online]. [cit. 2015-11-30]. DOI: 10.1.1.114.3843. Dostupné z: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.3843*

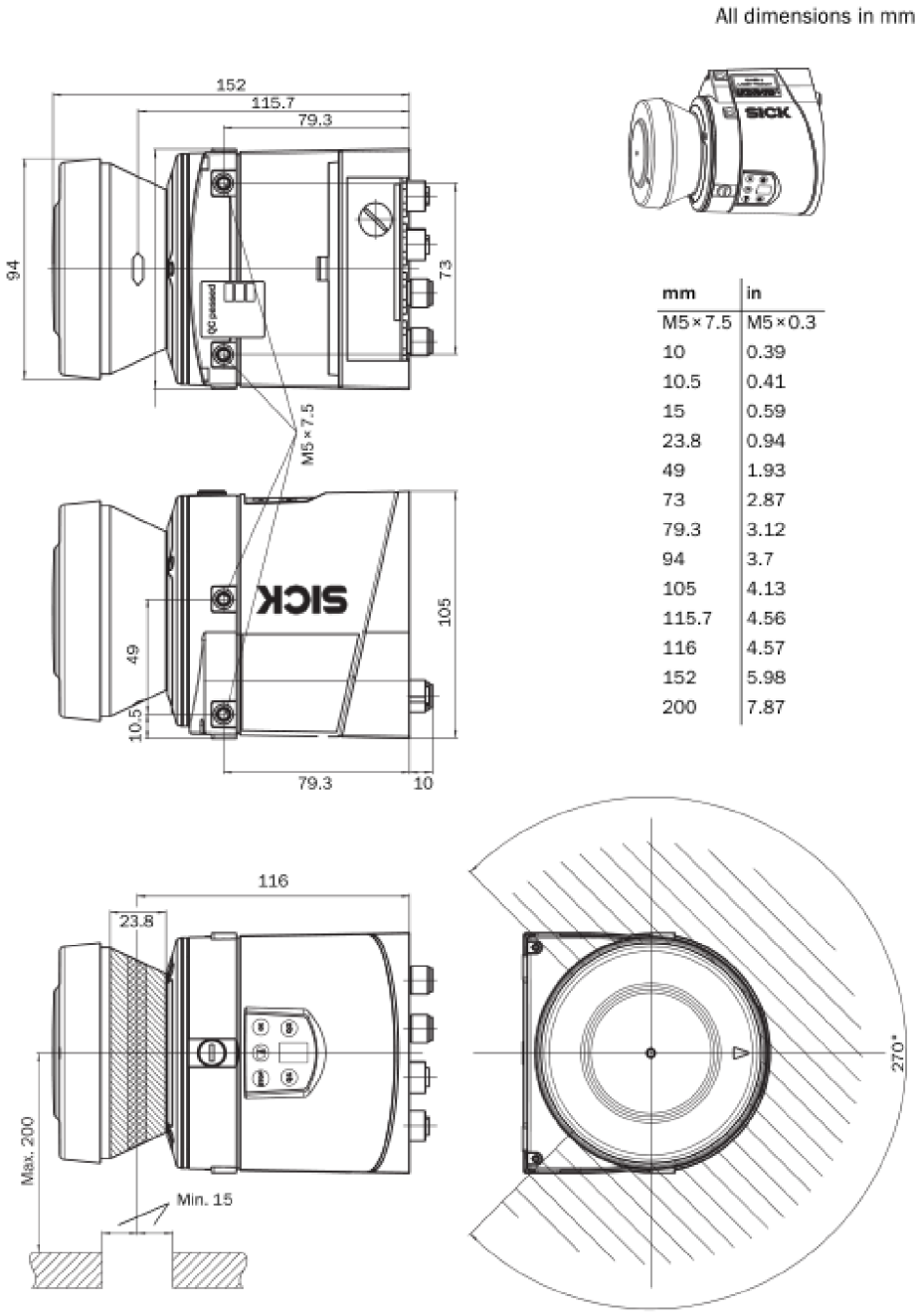
[24] ANTECKÝ, J. Aplikace metody zarovnání snímku pro lokalizaci autonomního robotu. *Master's thesis,* Vysoká škola bánská - Technická univerzita Ostrava (2014).

[25] KONEČNÝ, J. Principles of mobile service robots. *Dissertation thesis,* Vysoká škola báňská - Technická univerzita Ostrava (2014).

[26] SICK AG. LMS SICK 100: Laserový senzor [online]. [cit. 2015-09-26]. Dostupné z: *http://www.hizook.com/files/publications/SICK_LMS100.pdf*

[27] Principy programování: Robustnost a korektnost. Zive.cz [online]. Praha: Mladá fronta a.s., 2009 [cit. 2016-01-16]. Dostupné z: *http://principyprogramovani.blog.zive.cz/2009/09/robustnost-a-korektnost/*

Seznam příloh



mm	in
M5 × 7.5	M5 × 0.3
10	0.39
10.5	0.41
15	0.59
23.8	0.94
49	1.93
73	2.87
79.3	3.12
94	3.7
105	4.13
115.7	4.56
116	4.57
152	5.98
200	7.87

Technická dokumentace senzoru. [26]